Homework Assignment--NextDate-- test case design

by:
Abdullah Hanoosh (100749026)

Github:

1- follow our algorithm to design test cases and group them in a test case table (chapter 6 of the book). Highlight characteristics, domains, blocks, values. Use the pairwise coverage criteria.

Test Case #	**Day**	**Month**	**Year**	**Expected Output**	**Additional Info**
TC1	28	2	1900	1-3-1900	Non-leap year February end
TC2	29	2	2000	1-3-2000	Leap year February end
TC3	30	4	2021	1-5-2021	April end
TC4	31	7	2021	1-8-2021	July end
TC5	31	12	2212	1-1-2213	Year boundary
TC6	15	1	1812	16-1-1812	Normal day
TC7	29	2	1900	Invalid	Non-leap year February
TC8	32	1	2020	Invalid	Invalid day
TC9	31	4	2021	Invalid	Invalid April end
TC10	1	13	2021	Invalid	Invalid month

Characteristics and Domains:

Day= 1-31		Month= 1-12	Year = 1812-2212

Domains and Blocks

Days valid = 1-28 for all months, 29 (for all months except Feb on non-leap years), 30 (for April, June, September, and November), 31 (for January, March, May, July, August, October, and December)

Days invalid = 0, and all numbers greater than 31

Months valid = 1-12

Months invalid = 0, and all numbers greater than 12

Year valid = 1812-2212

Year invalid = less than 1812, greater than 2212

Values for Testing

Day: 1,15,28,29,30,31

Month: 1,2,4,7,12

Year: 1812, 1900, 2000, 2212

Pairwise Coverage

Pair 1:

Pairs the day and months together to ensure the testing of non-leap years (28 Feb), leap years (29 Feb), as well as the testing of 30 or 31 days in a month (30 April and 31 July)

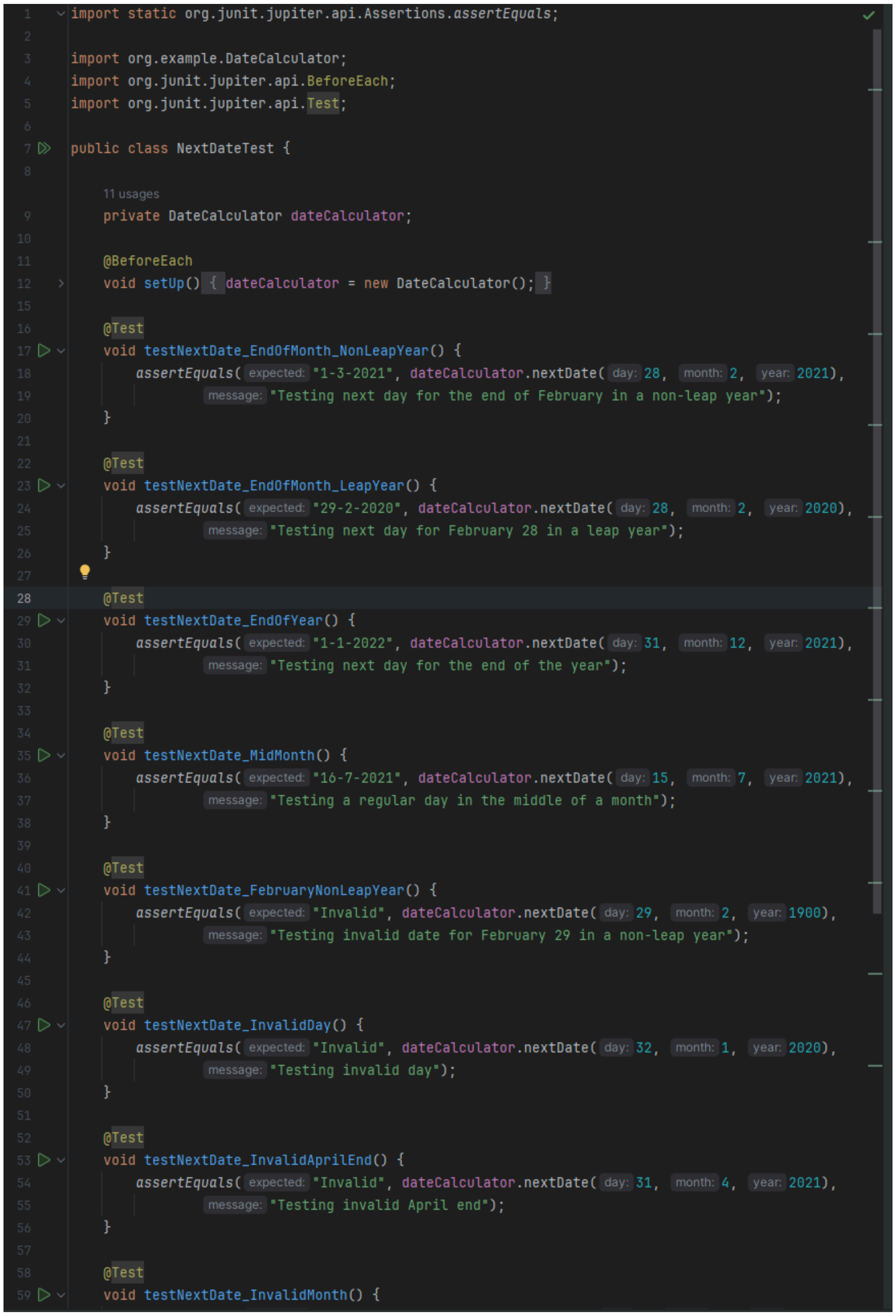
Pair 2:

Pairs the months and years together to test for Feb. in leap years and non-leap years (Feb 1900 and Feb 2000)

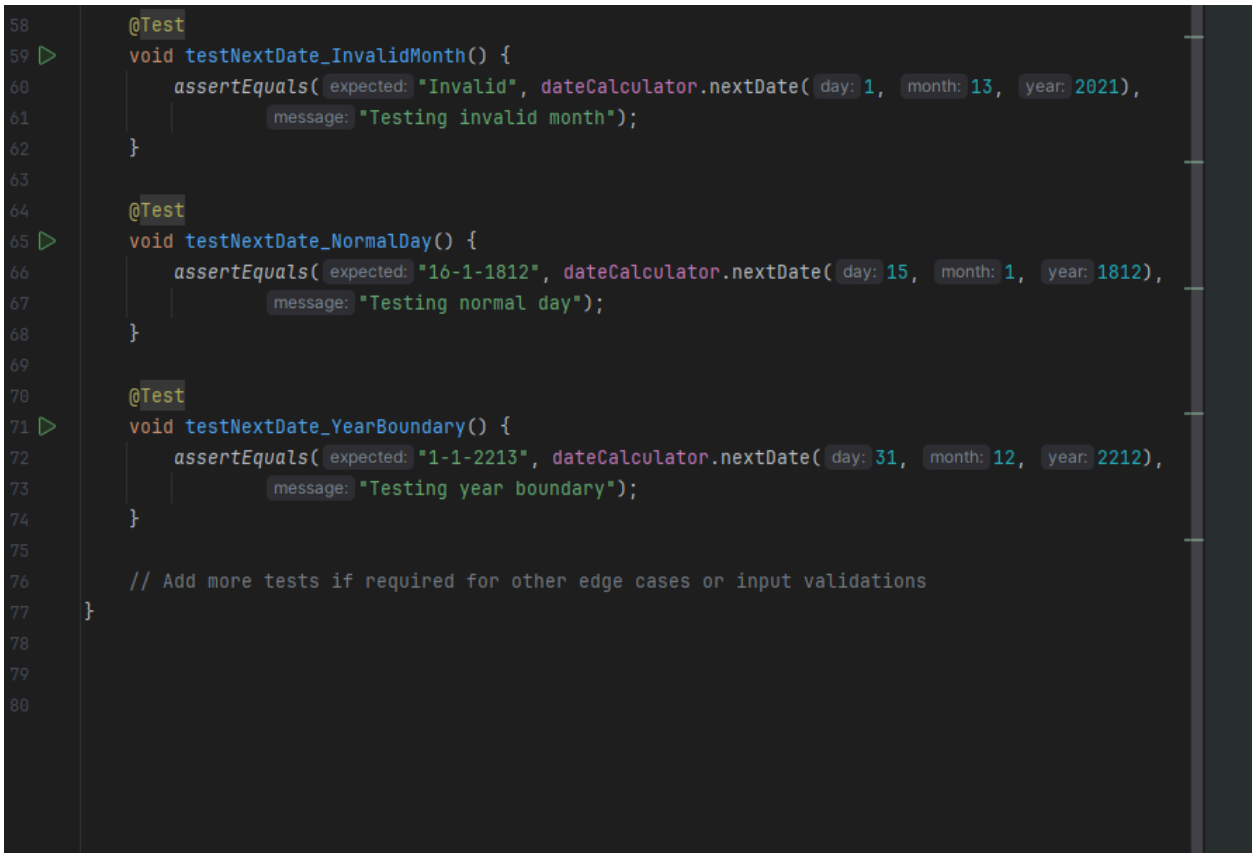
Pair 3:

Pairs the days and years together for boundary testing (31 Dec 2212)

2- write a Junit test script (This is a chance to experiment with different Junit annotations to organize your test cases)

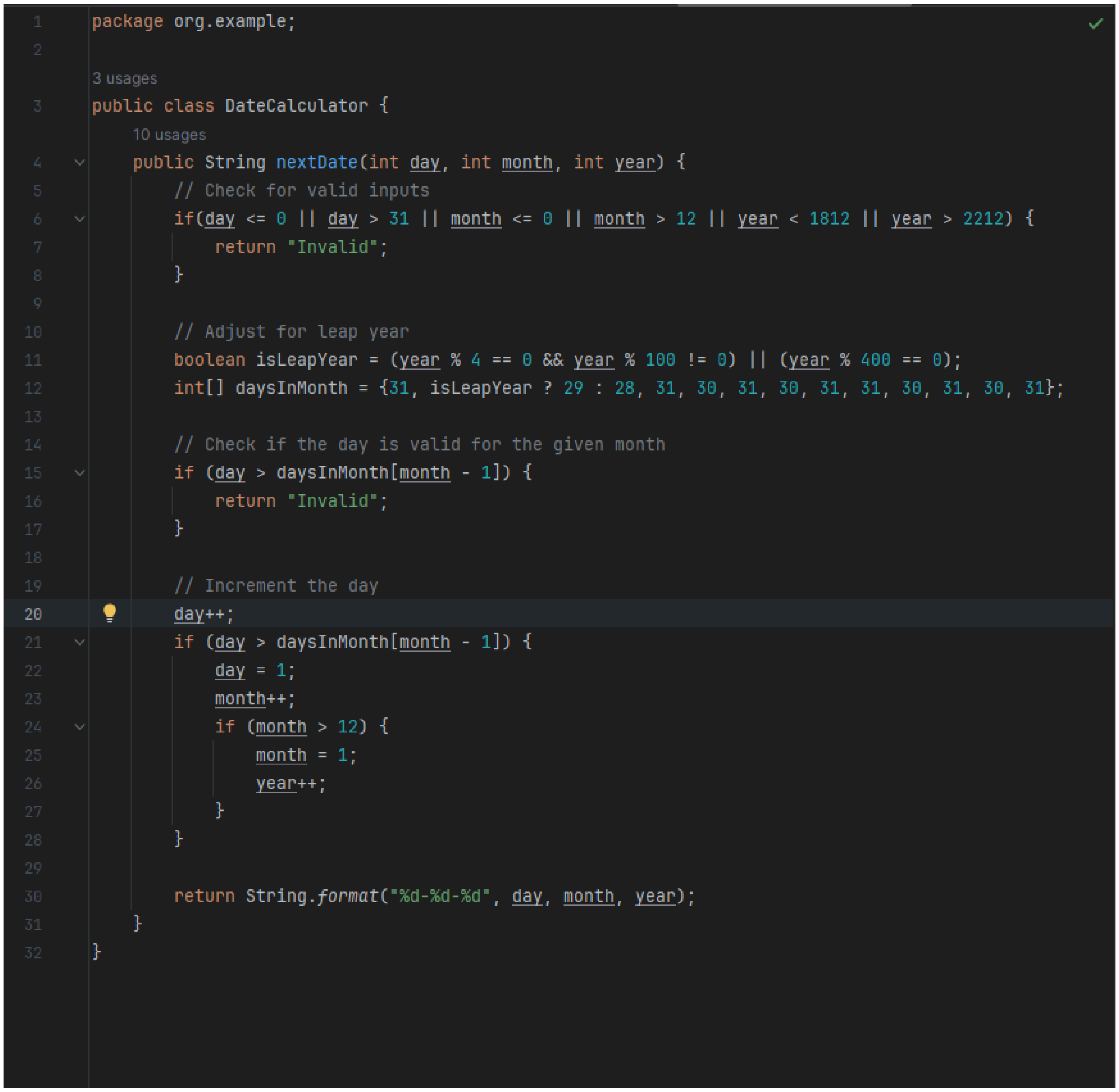


```java
import static org.junit.jupiter.api.Assertions.assertEquals;

import org.example.DateCalculator;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

public class NextDateTest {

    11 usages
    private DateCalculator dateCalculator;

    @BeforeEach
    void setUp() { dateCalculator = new DateCalculator(); }

    @Test
    void testNextDate_EndOfMonth_NonLeapYear() {
        assertEquals( expected: "1-3-2021", dateCalculator.nextDate( day: 28,  month: 2,  year: 2021),
                 message: "Testing next day for the end of February in a non-leap year");
    }

    @Test
    void testNextDate_EndOfMonth_LeapYear() {
        assertEquals( expected: "29-2-2020", dateCalculator.nextDate( day: 28,  month: 2,  year: 2020),
                 message: "Testing next day for February 28 in a leap year");
    }

    @Test
    void testNextDate_EndOfYear() {
        assertEquals( expected: "1-1-2022", dateCalculator.nextDate( day: 31,  month: 12,  year: 2021),
                 message: "Testing next day for the end of the year");
    }

    @Test
    void testNextDate_MidMonth() {
        assertEquals( expected: "16-7-2021", dateCalculator.nextDate( day: 15,  month: 7,  year: 2021),
                 message: "Testing a regular day in the middle of a month");
    }

    @Test
    void testNextDate_FebruaryNonLeapYear() {
        assertEquals( expected: "Invalid", dateCalculator.nextDate( day: 29,  month: 2,  year: 1900),
                 message: "Testing invalid date for February 29 in a non-leap year");
    }

    @Test
    void testNextDate_InvalidDay() {
        assertEquals( expected: "Invalid", dateCalculator.nextDate( day: 32,  month: 1,  year: 2020),
                 message: "Testing invalid day");
    }

    @Test
    void testNextDate_InvalidAprilEnd() {
        assertEquals( expected: "Invalid", dateCalculator.nextDate( day: 31,  month: 4,  year: 2021),
                 message: "Testing invalid April end");
    }

    @Test
    void testNextDate_InvalidMonth() {
```



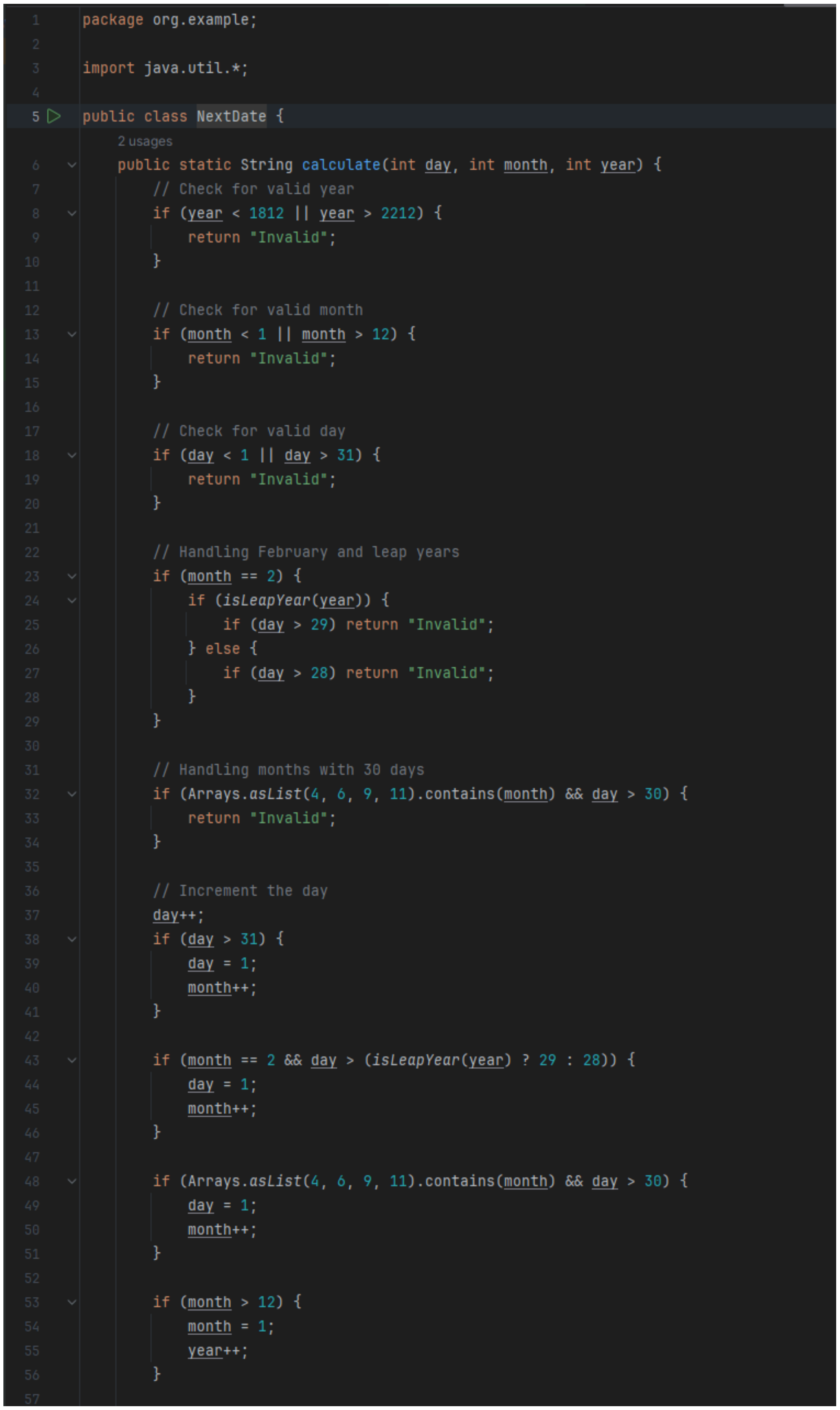
```java
    @Test
    void testNextDate_InvalidMonth() {
        assertEquals( expected: "Invalid", dateCalculator.nextDate( day: 1,  month: 13,  year: 2021),
                 message: "Testing invalid month");
    }

    @Test
    void testNextDate_NormalDay() {
        assertEquals( expected: "16-1-1812", dateCalculator.nextDate( day: 15,  month: 1,  year: 1812),
                 message: "Testing normal day");
    }

    @Test
    void testNextDate_YearBoundary() {
        assertEquals( expected: "1-1-2213", dateCalculator.nextDate( day: 31,  month: 12,  year: 2212),
                 message: "Testing year boundary");
    }

    // Add more tests if required for other edge cases or input validations
}
```

3- write the corresponding java code
DateCalculator.java

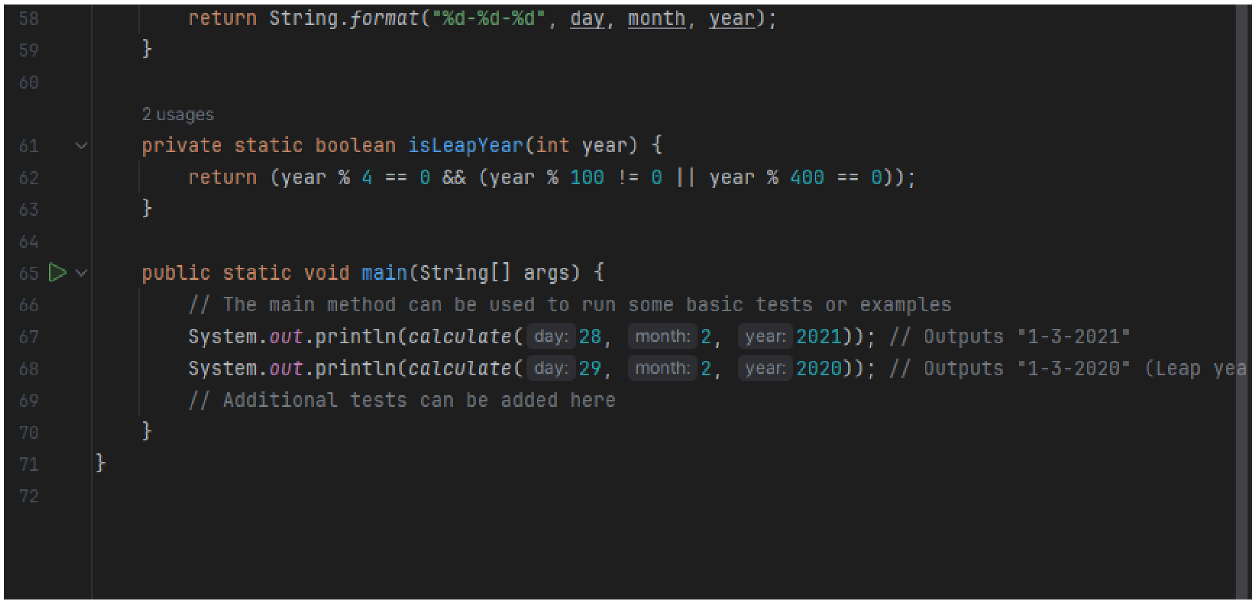


```java
package org.example;

3 usages
public class DateCalculator {
    10 usages
    public String nextDate(int day, int month, int year) {
        // Check for valid inputs
        if(day <= 0 || day > 31 || month <= 0 || month > 12 || year < 1812 || year > 2212) {
            return "Invalid";
        }

        // Adjust for leap year
        boolean isLeapYear = (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
        int[] daysInMonth = {31, isLeapYear ? 29 : 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

        // Check if the day is valid for the given month
        if (day > daysInMonth[month - 1]) {
            return "Invalid";
        }

        // Increment the day
        day++;
        if (day > daysInMonth[month - 1]) {
            day = 1;
            month++;
            if (month > 12) {
                month = 1;
                year++;
            }
        }

        return String.format("%d-%d-%d", day, month, year);
    }
}
```

NextDate.java



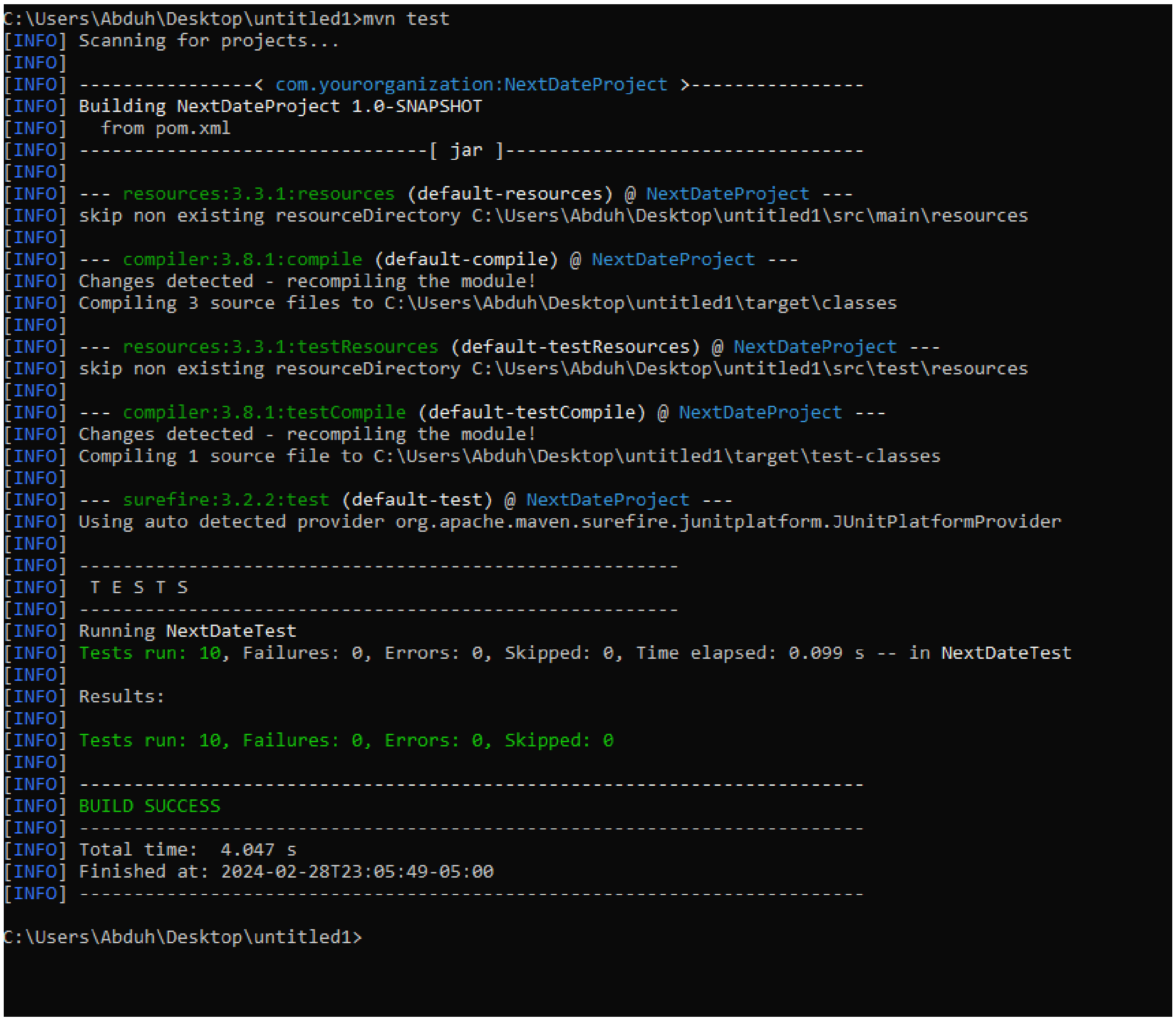
```java
package org.example;

import java.util.*;

public class NextDate {
    2 usages
    public static String calculate(int day, int month, int year) {
        // Check for valid year
        if (year < 1812 || year > 2212) {
            return "Invalid";
        }

        // Check for valid month
        if (month < 1 || month > 12) {
            return "Invalid";
        }

        // Check for valid day
        if (day < 1 || day > 31) {
            return "Invalid";
        }

        // Handling February and leap years
        if (month == 2) {
            if (isLeapYear(year)) {
                if (day > 29) return "Invalid";
            } else {
                if (day > 28) return "Invalid";
            }
        }

        // Handling months with 30 days
        if (Arrays.asList(4, 6, 9, 11).contains(month) && day > 30) {
            return "Invalid";
        }

        // Increment the day
        day++;
        if (day > 31) {
            day = 1;
            month++;
        }

        if (month == 2 && day > (isLeapYear(year) ? 29 : 28)) {
            day = 1;
            month++;
        }

        if (Arrays.asList(4, 6, 9, 11).contains(month) && day > 30) {
            day = 1;
            month++;
        }

        if (month > 12) {
            month = 1;
            year++;
        }

```



```java
        return String.format("%d-%d-%d", day, month, year);
    }

    2 usages
    private static boolean isLeapYear(int year) {
        return (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0));
    }

    public static void main(String[] args) {
        // The main method can be used to run some basic tests or examples
        System.out.println(calculate( day: 28,  month: 2,  year: 2021)); // Outputs "1-3-2021"
        System.out.println(calculate( day: 29,  month: 2,  year: 2020)); // Outputs "1-3-2020" (Leap yea
        // Additional tests can be added here
    }
}
```

4- run the tests and debug if need be.

5- repeat until the quality of the code is assured



```
C:\Users\Abduh\Desktop\untitled1>mvn test
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------< com.yourorganization:NextDateProject >----------------
[INFO] Building NextDateProject 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ NextDateProject ---
[INFO] skip non existing resourceDirectory C:\Users\Abduh\Desktop\untitled1\src\main\resources
[INFO]
[INFO] --- compiler:3.8.1:compile (default-compile) @ NextDateProject ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 3 source files to C:\Users\Abduh\Desktop\untitled1\target\classes
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ NextDateProject ---
[INFO] skip non existing resourceDirectory C:\Users\Abduh\Desktop\untitled1\src\test\resources
[INFO]
[INFO] --- compiler:3.8.1:testCompile (default-testCompile) @ NextDateProject ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to C:\Users\Abduh\Desktop\untitled1\target\test-classes
[INFO]
[INFO] --- surefire:3.2.2:test (default-test) @ NextDateProject ---
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running NextDateTest
[INFO] Tests run: 10, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.099 s -- in NextDateTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 10, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  4.047 s
[INFO] Finished at: 2024-02-28T23:05:49-05:00
[INFO] ------------------------------------------------------------------------

C:\Users\Abduh\Desktop\untitled1>
```